

AL-TP-1991-0058

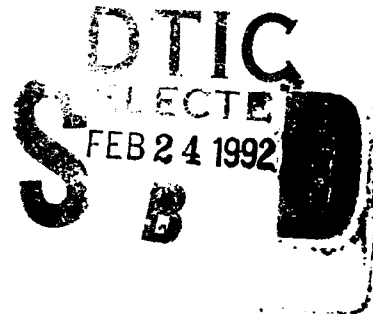
AD-A246 459



**AIR COMBAT MANEUVERING EXPERT
SYSTEM TRAINER**

Robert J. Bechtel

**Merit Technology, Incorporated
5068 West Plano Parkway
Plano, TX 75093**



**HUMAN RESOURCES DIRECTORATE
AIRCREW TRAINING RESEARCH DIVISION
Williams Air Force Base, AZ 85240-6457**

January 1992

Final Technical Paper for Period July 1988 - December 1991

Approved for public release; distribution is unlimited.

92 2 19 089

92-04392



**AIR FORCE SYSTEMS COMMAND
BROOKS AIR FORCE BASE, TEXAS 78235-5000**

**ARMSTRONG
LABORATORY**

NOTICES

This technical paper is published as received and has not been edited by the technical editing staff of the Armstrong Laboratory.

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

The Office of Public Affairs has reviewed this paper, and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nationals.

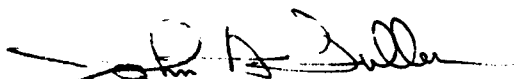
This paper has been reviewed and is approved for publication.



RICHARD A. THURMAN
Contract Monitor



DEE H. ANDREWS, Technical Director
Aircrew Training Research Division



JOHN H. FULLER, JR., Colonel, USAF
Chief, Aircrew Training Research Division

REPORT DOCUMENTATION PAGEForm Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE January 1992	3. REPORT TYPE AND DATES COVERED Final - July 1988 - December 1991	
4. TITLE AND SUBTITLE Air Combat Maneuvering Expert System Trainer			5. FUNDING NUMBERS C - F33615-88-C-0011 PE - 62205F PR - 1123 TA - 35 WU - 11	
6. AUTHOR(S) Robert J. Bechtel				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Merit Technology, Incorporated 5068 West Plano Parkway Plano, TX 75093			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Armstrong Laboratory Human Resources Directorate Aircrew Training Research Division Williams Air Force Base, AZ 85240-6457			10. SPONSORING/MONITORING AGENCY REPORT NUMBER AL-TP-1991-0058	
11. SUPPLEMENTARY NOTES Armstrong Laboratory Technical Monitor: Dr. Richard Thurman, (602) 988-6561				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) Models of decision making can be useful in a number of applications. Building, validating, and creating the uses for such models is usually a significant undertaking for any domain of interesting size. The Air Combat Maneuvering Expert System Trainer (ACMEST) project used knowledge acquisition techniques developed in artificial intelligence to construct models of pilot tactical decision making, and to implement those models as rule-based inference system. The model implementations were then used in a man-in-the-loop simulation environment to teach air combat maneuvering. The resulting system serves as a proof-of-concept demonstrator, but must have greater flexibility in knowledge representation, validated models, and improvements in display capability to become useful in operational training.				
14. SUBJECT TERMS Air combat maneuvering Decision making Expert model			ITS Rule-based inference Simulation-based training Tactical decision making	
			15. NUMBER OF PAGES 44	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

CONTENTS

	PAGE
SUMMARY	1
BACKGROUND	2
Scope of Effort	2
Risk Areas	3
APPROACH	5
System Knowledge	5
Program Description	7
Software Reuse	11
Display Implementation	14
Simulation Modeling	13
Inference Software	13
Interprocessor Communication	16
Multitasking and Interprocessor Communication on the PC	16
DISCUSSION	17
Pedagogic Issues	19
Role in Pilot Training	19
Structured Training vs Free Fight	20
Graphics and User Interface Issues	22
Single Fixed Viewpoint/Narrow Aperture	22
Interaction/Mediation Structure	23
Simulation Fidelity and Display Fidelity Issues	23
Suggestions for Future Work	24
REFERENCES	26
APPENDIX: Chronology	27

LIST OF FIGURES

FIGURE	PAGE
1 ACMEST Processing	7
2 Instructional Expert	8
3 System Architecture	10

PREFACE

The Air Combat Maneuvering Expert System Trainer (ACMEST) is one of four parallel efforts performed under a Program Research and Development Announcement (PRDA) to investigate the development and use of expert models of pilot knowledge in air combat. ACMEST used a traditional knowledge engineering approach to create the pilot models and focused its efforts on using the models to provide instruction.

This research represents a portion of research and development at the Armstrong Laboratory Human Resources Directorate, Aircrew Training Research Division (AL/HRA). This effort supports one of the laboratory's goals to increase combat readiness and job performance by developing and demonstrating more cost-effective ways of acquiring and maintaining new skills.

This work was conducted under Work Unit 1123-35-11, Air Combat Maneuvering Expert System Trainer. Work was completed by Merit Technology Incorporated under Contract F33615-88-C-0011. Dr Richard A. Thurman, AL/HRAU, was the work unit and contract monitor for this effort.

AIR COMBAT MANEUVERING EXPERT SYSTEM TRAINER

SUMMARY

Armstrong Laboratory Human Resources Directorate, Aircrew Training Research Division (AL/HRA) is investigating the creation and use of models of pilot expertise in air combat maneuvering. To support this effort, AL/HRA contracted with Merit Technology Incorporated to construct an Air Combat Maneuvering Expert System Trainer (ACMEST).

ACMEST is a collection of computer programs executed on an advanced PC/AT-architecture computer with an Intel 80386 processor and a Silicon Graphics IRIS workstation. The computer programs were developed as an exploration and demonstration of the feasibility of applying model-based intelligent training techniques to air combat.

The system contains rule-based models of air combat expertise for an instructor pilot, an adversary, and the student pilot. These models are used in conjunction with a man-in-the-loop flight simulation to detect flaws in student technique and to suggest alternate courses of action.

While the completed ACMEST system demonstrates the feasibility of applying intelligent training techniques to air combat, it is little more than a prototype. As currently implemented, the system is severely limited by its display capabilities, and less severely limited by the choice of an exclusively rule-based approach to knowledge representation and modeling.

Future efforts should focus on overcoming the shortfalls of this system and on incorporating the work in pilot modeling and debriefing conducted under parallel efforts.



Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Avail and/or	
Dist	Special
A-1	

BACKGROUND

The ACMEST system grew out of a Program Research and Development Announcement (PRDA) which appeared in the Commerce Business Daily of 2 January 1988. That announcement stated "The purpose of this effort is to develop and validate an expert model of pilot decision making in air combat maneuvering (ACM). ... The system shall be capable of providing ACM decision-making training at the transition or continuation level ..." Merit Technology Incorporated proposed construction of an intelligent tutoring system (ITS) containing rule-based models of pilot and instructor decision making coupled with computational flight simulation and high-speed, moderate fidelity graphic displays to achieve the training capability sought in the announcement.

While there was a growing body of knowledge in the general area of ITS, there were no specific systems which attempted automated instruction in the ACM domain, and there were few systems, if any, which provided intelligent instruction in any domain which required motor responses in "real time" measured in fractions of a second. In some sense then, our task was to explore whether such a system was feasible, using prototype construction as our approach.

As the recipient of one of four awards under the PRDA, Merit has constructed and demonstrated the ITS described in this paper. As is often the case in research efforts, our original plans were modified as we progressed to keep our focus on the development of a complete simulation-based capability. The changes from the original plan have caused some of our initial goal to remain unfulfilled, but have also led to other capabilities beyond our starting intent.

Scope of Effort

The ACMEST effort required design, implementation (or modification to existing software) and integration of the following components:

- a dynamic, man-in-the-loop, real-time simulation with one pilot flight control over realistic fighter aircraft models and autonomous goal-directed (computer) control of adversary fighter aircraft models (models to include realistic flight equations, sensors and weapons);
- graphics simulations with displays capable of indicating the relative position and geometry of up to three aircraft simultaneously from any perspective

(including cockpit out-the-window displays with realistic head-up display (HUD) symbology);

- a knowledge-based ACM tactics planner capable of integrating the current situation, training objectives and an expert model of offensive and defensive pilot decision making to determine the best course of action for the student pilot to follow at any given point in time during the simulated mission; and
- a computer-aided instruction (CAI) subsystem capable of comparing the pilot-selected and "school" courses of action, ascertaining the differences, and preparing a training plan of on-line, scenario replay and post-flight tutorial material.

In addition, the effort was planned to include a series of demonstrations of these capabilities in a laboratory environment with actual pilot subjects.

The developed ACMEST system was implemented on the hardware and software fully compatible with the ACM performance measurement system (PMS) at Luke AFB, AZ.

Risk Areas

As we started the project, Merit perceived five distinct risk areas which would require evaluation, management, and possible mitigation to achieve our goal. The first two areas were very closely related and involved the acquisition and representation of expert knowledge of air combat.

Transferring knowledge of air combat from both documentary and human sources into computer software was the primary risk area jeopardizing the successful attainment of ACMEST objectives. While our previous experience on other programs gave us confidence that we could construct simple models, there was a possibility that the available techniques for knowledge acquisition and transfer would break down at more sophisticated levels of modeling.

Based on those earlier, simpler models, it was clear that a much "deeper" representation of factors affecting air combat would be required for success in teaching. A model concerned purely with producing "pilot-like" behavior could demonstrate acceptable performance using only superficial aspects of a tactical situation and, in our experience, could ignore many or all issues of situation evolution over time. A model capable of providing explanations that would be meaningful to trainee pilots (rather than computer science or psychology researchers) as well as appropriate behaviors

needed an ability to examine the factors leading to decisions, and to trace those factors back to deeply grounded principles of aerodynamics and tactical doctrine. Such multilevel models had not been widely used, and the successful development and use of internal computer representations for a layered modeling approach was our second risk area.

The third risk area was validation of the expert systems (models) developed. If the expert models are to be used in an instructional setting as we have employed them, it is important that the user community have confidence that what is being taught is both accurate and useful. Validation serves to provide such confirmation. Unfortunately, general-purpose validation of rule-based expert systems is an open research area. We have had to settle for informal, qualitative judgements by experienced personnel regarding the accuracy and utility of this system, making validation one of the weaker aspects of our effort.

The fourth risk area was finding a proper balance between artificial intelligence and more conventional algorithmic solutions. "If the only tool you have is a hammer, the whole world looks like a nail." Given the background of most team members in artificial intelligence (AI) and the general tenor of our original proposal, we had to maintain a constant watch to ensure that we did not unnecessarily complicate the system by using newly-developed, esoteric techniques to accomplish what long-standing, well-understood algorithms could do. By the same token, it was important to recognize as early as possible when conventional approaches were simply inadequate, so that alternatives could be devised, rather than having to limp along under limitations imposed by existing algorithms.

The fifth risk area was ultimate system throughput and real-time performance. We had proposed (and executed) an approach which relied on the use of rule-based inference techniques for expert modeling. The general perception has been that these techniques impose a significant burden on the available computational resource causing performance problems. Since our approach required a high simulate update rate and significant display bandwidth, there was a real question as to whether we could execute the models quickly enough to avoid delays in simulation and display.

APPROACH

System Knowledge

The ACMEST system incorporates two distinct but interrelated types of intelligence: (a) knowledge about aircraft and ACM to impart to the pilot, and (b) knowledge about teaching. The knowledge of ACM embodied in the ACMEST expert system had to be augmented with a clear, concise, timely method of imparting it to the pilot. The ACMEST system also required the capability to solve problems independently as well as critique student solutions.

Merit's approach hinges on the ability to recognize the crucial elements of knowledge which are required for successful air combat maneuvering, then to capture and encode those elements in a high-performance expert system. These elements of knowledge include, but are not limited to:

*** Hardware (both friendly and enemy)**

- Aircraft (performance capabilities, size/visibility/detectability, etc.)
- Weapons system capabilities...including sensors, weapons onboard, reliability, etc.
- Numbers of aircraft involved
- Electronic countermeasures/infrared countermeasures (ECM/IRCM) capabilities

*** Environment**

- Location...how far and what direction from base/friendly airspace (both friendly and enemy)
- Ground threat
- Sun position
- Cloud locations
- Visibility
- Electromagnetic environment...presence of radar/communications jamming, etc.
- Likelihood of additional friendly or enemy support entering the fight...more fighters, etc.
- Ground-controlled interceptor (GCI) support (both friendly and enemy)
- Airborne tanker support (both friendly and enemy)

*** Situation state (both friendly and enemy)**

- **Airspeed...both absolute and in relation to aircraft performance capabilities**
- **Altitude...both absolute and in relation to aircraft performance capabilities**
- **Attitude...pitch and roll orientation of aircraft**
- **Aircraft response...how is the aircraft responding to control inputs**
- **Weapons system/sensor status...how are things working, radar lock-on, etc.**
- **ECM/Radar heading and attitude warning (RHAW) status/inputs...are we targeted, etc.**
- **Geometry...relative positions of all aircraft**
- **Fuel state...both absolute and relative to what can be done with it (i.e., how long before it will be necessary to disengage in order to reach safety**
- **Offensive/defensive status...are we offensive or defensive, where is each aircraft in relation to its firing or vulnerable envelopes, etc.**
- **Progress of fight...are we getting relatively more or less offensive...how fast is this status changing...how long before roles will change or one aircraft will enter weapons envelope, etc.**
- **Pilot mindset...is the pilot in an aggressive or defensive state of mind**

*** Pilot Capabilities (both friendly and adversary)**

- **Skill level...how good is each pilot at performing the current task, what training has been received, experience, etc.**
- **Proficiency level...how proficient is each pilot at performing the current task, how long since each task was practiced, etc.**

*** Mission/Intent (both friendly and adversary)**

- **Mission...what is the mission of each aircraft (i.e., does the pilot want to engage or avoid engagement, is the pilot defending a point or seeking to control airspace, is the pilot trying to bomb a target or trying to shoot another aircraft, etc.)**
- **Intent...what are each pilot's selected tactics, what are they trying to do (i.e., kill, scare, disengage, etc.)**

*** Tactics**

- **What "family" of tactics (i.e., angles tactics or energy tactics) are appropriate under the current circumstances**
- **What weapon envelope should the pilot attempt to satisfy (i.e., what should be the primary weapon if there is a choice)**
- **What particular tactic is called for at the current time (i.e., level turn, hi yo-yo, zoom climb, etc.)**

- What maneuver is necessary to accomplish the desired tactic (i.e., power, bank angle, G level, aircraft configuration, switch selection, etc.)

In the course of development of ACMEST, Merit was able to include some, but not all, of these knowledge types in the system. We made simplifying assumptions which had minimal impact on the central function of the system. For example, we did not consider missions in which one of the combatants is configured for air-to-surface attack because that was not the principal focus of the system.

Program Description

The basic ACMEST process is best visualized as a pilot-centered loop, as shown in Figure 1. From the computer displays, the pilot receives visual information. Usually, this information is a real-time, out-the-window view from the simulated aircraft during combat. Reacting to this information, the pilot manipulates the controls (throttle and stick joysticks). The changed positions of the controls are detected by both the simulation software and the instructional expert.

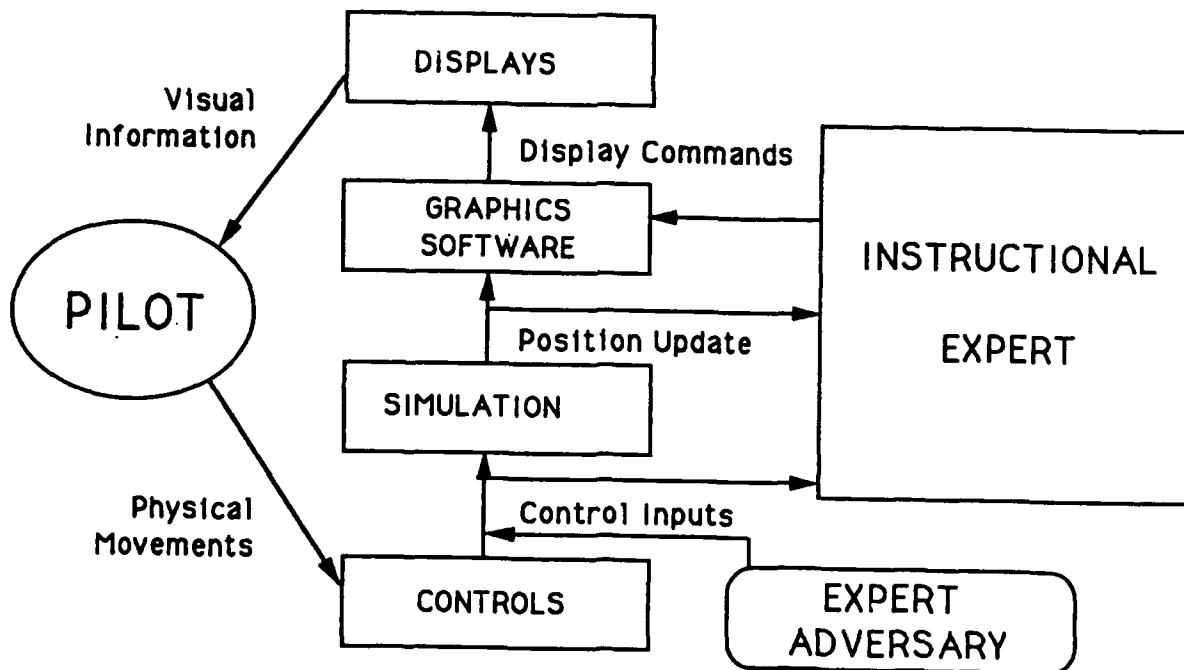


Figure 1
ACMEST PROCESSING

The simulation software takes the changed control inputs and generates position updates based on a six-degree-of-freedom (6DOF) flight model. If missiles are in flight, their position is updated using a three-degree-of-freedom (3DOF) flight model. The updated positions are passed to the graphics display software which calculates visibility and apparent orientation and issues commands to the display hardware, which redraws to update the pilot on the situation.

As described so far, the system is a fairly straightforward, moderate fidelity, man-in-the-loop flight simulation system. Two additional subsystems extend the capability to ACM training.

The first additional capability is termed the "expert adversary." This is a computerized behavioral model of a skilled pilot. The model is implemented as a collection of inference rules, sometimes called an "expert system." The adversary model controls a second simulated aircraft which flies as an opponent to the human student pilot. The adversary receives inputs from the simulation describing the current position and orientation of the adversary aircraft, and as much information about the student's aircraft as could be gleaned from visual sighting or available sensors. Working from this information, the adversary model generates control inputs to the simulation. Basically, the expert adversary represents a parallel flight simulation loop which shares the simulation software, and which does not require a separate graphic display or physical joysticks.

The second additional capability is the instructional expert, which is roughly analogous to an instructor. The inner workings of the instructional expert are shown in Figure 2 and are built around the comparison of behavior.

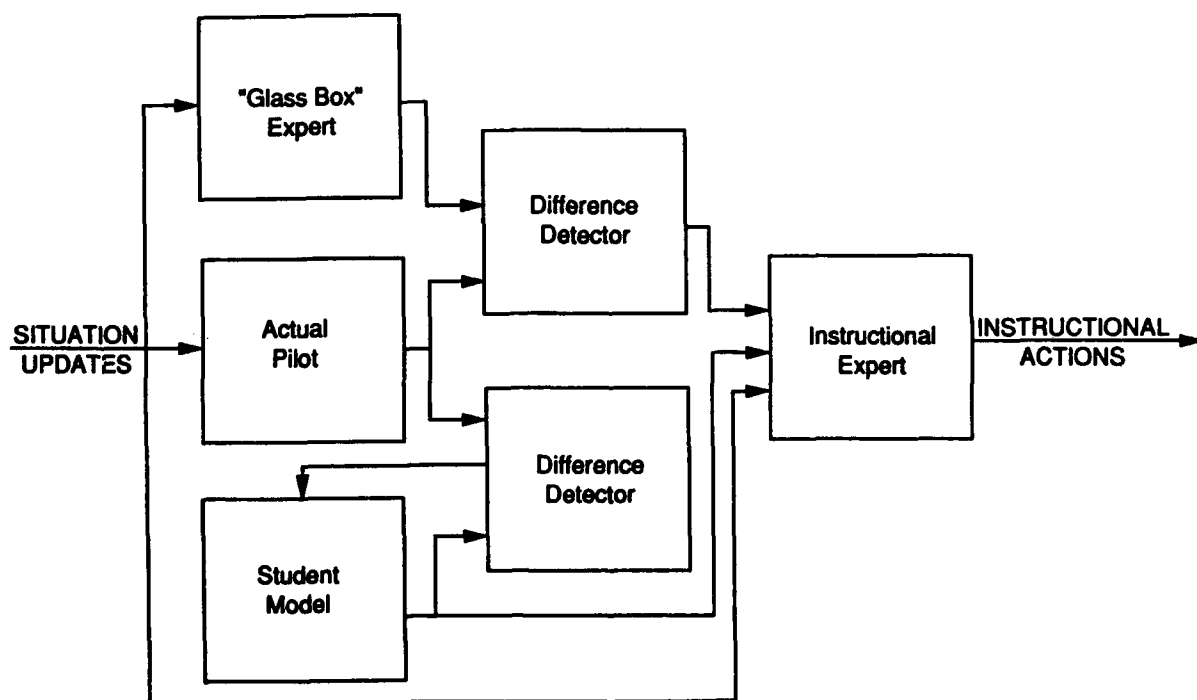


Figure 2
INSTRUCTIONAL EXPERT

As mentioned earlier, the instructional expert receives the control inputs from the student pilot. It also receives situation descriptions (in the form of positional updates) from the simulation. Within the instructional expert, there are three distinct models of expertise.

The first is called the "glass box" expert. This is a behavioral model of ACM competence, which is similar (or in some cases identical) to the adversary model. The glass box expert knows what to do in any air combat situation and can produce appropriate control settings from a situation description. It is called a "glass box" model because its interior must be open and available for inspection. That is, processes external to the model must be able to trace the reasoning which leads from situation to action, rather than simply accepting outputs from inputs as is the case with a "black box" model. The reasoning trace will be used in the instructional process.

The current situation is presented to the glass box expert model which produces a set of recommended "correct" control settings as output. These control settings are compared to those chosen by the student pilot and any differences are noted for later use. It should be noted that a difference between the settings chosen by the glass box expert and the student does not necessarily indicate an error by the student -- many situations in air combat admit more than a single approach. However, the divergence must be recorded for analysis so that if it does indicate a less than ideal choice by the student, the system can intervene if appropriate.

The second model captures the system's current understanding of the student's knowledge. A student who is highly skilled at air combat should have a highly skilled model within the system. A student who has no skill at air combat should have a low- or no-skill model. The situation description is presented to the student model which produces a collection of control settings representing a prediction of what the student will do. This prediction is compared to the actual behavior of the student, just as the student's behavior was compared to the glass box expert. In this case, however, divergence between the student model and the actual student indicates that the student model is inaccurate, and it is used to suggest ways in which the model should be updated to more accurately reflect the student's current knowledge.

Like the glass box expert, the student model must be "transparent" and open to inspection by other processes. Instruction will proceed in part by consulting the student model to find weaknesses or deficiencies and presenting instructional opportunities which should permit the student to correct those shortcomings.

The third model captures instructional techniques and is (in the current system) the simplest of the three. Given a situation, a current snapshot of the student model and a comparison of actual pilot behavior with ideal pilot behavior, the instructional expert produces instructional actions, which can run the gamut from no current action through suspension of the simulation and presentation of tutorial text. Most frequently, the instructional action selected is the display of one or more "hints" to the student as to the proper course of action.

The processes involved in ACMEST are spread over two computers which are linked by an Ethernet local area network (LAN). The PMS facility at Luke AFB (the target destination for the system) offered a Silicon Graphics IRIS 2400 Turbo workstation computer, which was ideal for the high-speed graphic displays Merit planned to offer. Unfortunately, the central processing unit in that computer was not sufficient to handle both the graphics software and all the other computational requirements. For that reason, we introduced a second computer into our planned configuration. Based on the plans of the team from New Mexico State University to deliver a PC architecture computer based on the Intel 80386 chip, we decided to use that architecture as our second computer since one would be available at the Armstrong Laboratory Human Resources Directory, Aircrew Training Research Division (AL/HRA).

As shown in Figure 3, the IRIS workstation is responsible for all graphic displays and for limited flight data recorder functions. The simulation, the instructional expert, and the expert adversary all execute on the 386PC. In addition to the base processes, a communication process is required on each computer to handle the Ethernet connection between them.

TARGET IMPLEMENTATION

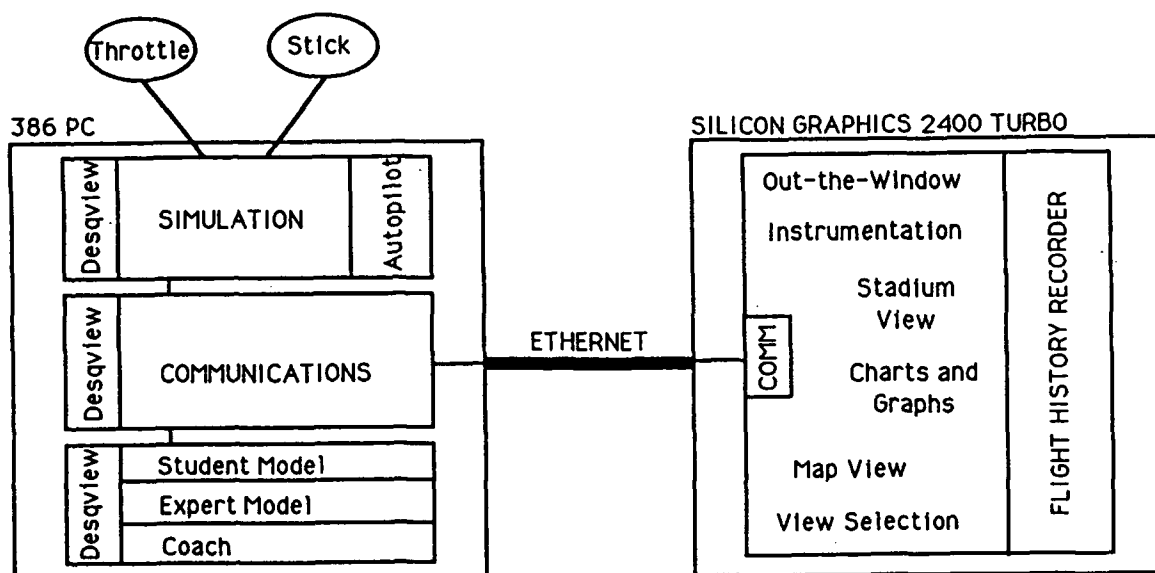


Figure 3
SYSTEM ARCHITECTURE

Software Reuse

Merit has made extensive use of existing software in the ACMEST effort. This software reuse has enabled us to do more in the areas of pilot modeling and ITS than would have been the case if we had needed to develop all of the software components in the system from scratch.

The display, simulation, inference, and interprocessor communications capabilities of ACMEST are provided by reused software. In addition, commercial off-the-shelf software permits the PC host to multitask (ordinarily not possible under Microsoft's Disk Operating System (MS-DOS) for the PC). The basic features and customization of each of these components is described in the following paragraphs.

Display Implementation

ACMEST generates its aircraft situation displays using a slightly modified version of a Merit-developed program called SeaScene. SeaScene generates three-dimensional, "out-the-window" views of a situation taking place in an ocean environment. Objects which the user may place into a situation include various types of ships, aircraft, missiles, etc. The user can use SeaScene to generate multiple views of a single situation or of several different situations. SeaScene is designed to be capable of generating several frames per second of a single view, allowing it to be used in conjunction with a real-time simulation.

The term "object" can be used to refer to aircraft, ships, missiles, etc. To avoid ambiguity, SeaScene differentiates between object classes and object instances. An object class exists in the Universe level of the database (see below). A class is a type of object (for instance, the F-18 object class describes how an F-18 aircraft should be drawn and what properties are possessed by all F-18 graphics objects). For each F-18 to be drawn, it is necessary to create an instance of the F-18 class. The F-18 instance is a member of the F-18 class, and also has its own properties, such as position and attitude. An object instance is defined as part of a situation (see below).

The SeaScene database has three levels which may be accessed by the user:

1. Universe - the entire SeaScene environment.
2. Situation - a collection of object instances which may be viewed together. Object instances in separate situations can never be drawn in the same view.
3. View - a specification of how to draw objects from a situation. This includes information such as viewpoint location, geographic area to be viewed, and location of view on the screen, etc. Each view is associated with a situation. Each situation may have multiple views defined simultaneously.

Note that for simple applications requiring only a single application and a single view, the user can ignore these distinctions since appropriate defaults are provided.

As an example, assume that the user desires to display out-the-window views of the Battle of Midway and the Battle of the Coral Sea simultaneously on the screen. For each battle, you want to display a view from an attacking aircraft and from an aircraft carrier being attacked.

To accomplish this, the user would first define two situations, one for each battle. For each situation, the user would then define instances of each object class corresponding to the players.

For example, in the Battle of Midway situation, the ships would be defined by creating two instances of class Aircraft Carrier for the ships Akagi and Kaga (This is just an example; SeaScene does not currently have aircraft carrier or dive bomber classes defined.). Similarly, several instances of object class SBD1 would be defined to represent the attacking dive bombers. After defining the ships and aircraft for Midway, the (separate) set of ships and aircraft for the Coral Sea situation would be similarly defined.

Once the situations have been defined, a set of views can then be defined for each situation. For example, a view in the Battle of Midway situation could be defined as being placed in the upper left quarter of the screen, with viewpoint attached to one of the SBD1 instances, and viewing along the aircraft axis. Another view could be simultaneously defined as being placed in the lower left quarter of the screen, attached to the object instance representing the Akagi, looking toward one of the SBD1 object instances.

All interactions with SeaScene are through commands issued to SeaScene. Commands may be issued either through data packets (suitable for use in a remote interface) or through function calls. In either case, the identical set of capabilities is available. In ACMEST, the commands to SeaScene are issued as data packets by the simulation software running on the PC. These data packets are sent over the Ethernet to the Silicon Graphics IRIS 2400 Turbo (or other IRIS display device) which is executing the SeaScene software.

Since all the commands to SeaScene in the ACMEST environment come as packets from the PC, SeaScene is able to maintain a record of all aircraft position and status information. This data is (optionally) stored in a simple "flight data recorder" which enables playback later from selected points.

SeaScene was originally developed under contract to the Naval Weapons Center, China Lake, CA, and all modifications required for ACMEST were performed under the ACMEST contract, thus ensuring the Government's full rights in the software. The primary modification to the system replaced the deep blue ocean surface with a brown "dirt" earth surface (leading to the informal name "DirtScene").

Simulation Modeling

Aircraft dynamics, weapon, and sensor simulation in ACMEST are provided by PC-based simulation software modeled after Merit's Battle Area Tactical Simulation (BATS) system. BATS was designed and developed by Merit to provide realistic, real-time simulation and analysis of complex avionics systems. BATS accommodates a variety of models to represent alternative configurations in gaming areas, aircraft types, flight dynamics, mission plans, route generation algorithms, threat laydowns, sensor suites, weaponry, fire doctrine, and tactics. New models can be easily introduced. BATS supports real-time interaction or batch jobs, automatic or man-in-the-loop or multiplayer gaming, and one-on-one, one-on-many, many-on-one, or many-on-many engagements.

Flight is realistically simulated over a flat earth with optional overlays of features such as threat sites. Through interaction with SeaScene, several sophisticated graphics options are available including:

- wide, out-the-window digital terrain views, with optional HUDs; and
- large-size plan views of the gaming area that show flight paths and relative aircraft locations.

The various displays are easily customized for specialized applications.

All Merit proprietary software has been removed from the simulation subsystem leaving full Government rights intact.

Inference Software

We have used MeriTool as the rule-based inference engine on the ACMEST project. MeriTool is a generic development and runtime software tool for knowledge-based systems developed by Merit Technology to support a wide variety of in-house and customer applications. MeriTool differs from most other knowledge-based system tools in that its functionality has grown from the needs of system developers in real-world

applications. As such, MeriTool encompasses a variety of techniques in knowledge representation and efficient inferencing which have proven essential in domains ranging from threat assessment and response to adaptive intelligence analysis. As the needs of customers have grown or changed, MeriTool has been customized to provide the most efficient functionality to meet their needs.

MeriTool is a toolkit for building embedded knowledge-based systems. In particular, it supports the creation and execution of rule-based inference systems. Some aspects of MeriTool's design and implementation are relevant to an effective use of its capabilities and are discussed here. Further general information on rule-based inference is widely available, typically in books such as those by Frost (1986).

Like virtually all rule-based inference systems, MeriTool has three components during operation:

1. a collection of rules (called the rule base),
2. a collection of facts (called the memory cache), and
3. an inference engine which applies the rules to the facts to reach conclusions.

MeriTool supports both common modes of rule-based inference: forward and backward. In forward (or data-driven) inference, changes (additions and deletions) are made to the memory cache, and the inference engine determines that the new memory cache state satisfies the conditions of any rules. If so, the inference engine carries out the actions called for by those rules, which typically involve modifying the memory cache, possibly triggering additional rules. This "retriggering" effect is called chaining, so forward inference is often called **forward chaining**.

Backward (or goal-driven) inference starts from a goal or fact whose truth is in question, and seeks to prove that there is a logical path connecting the available evidence with that goal. Typically, a backward inference approach starts by checking the memory cache to see if the goal is already known. If not, the inference engine checks the rules to see if any rules have the goal fact among their conclusions. If so, that rule can be used to conclude the goal fact, but only if its conditions are satisfied. Therefore, each of the conditions becomes a new goal. Again, the inference engine may link several rules together to create a chain which links the information available in the memory cache with the goal, leading to the term **backward chaining**.

MeriTool organizes the rule base into a network of tests to improve the speed of inference. The inference engine then passes memory cache changes (additions and deletions) through this network to determine what rules have satisfied conditions. The network representation and inference engine are loosely based on the RETE approach (Forgy, 1979), but have several enhancements which improve efficiency and support additional features, such as backward chaining.

Because the rule base is organized as a network, changes to the memory cache affect all rules simultaneously. This means that after any change, there may be several rules whose conditions are satisfied. Since MeriTool (ordinarily) executes on a serial machine, some sequence must be imposed on the application of the rule actions. This function is performed by the conflict resolution facility and is (to a degree) under user control. Understanding the operation of the conflict resolution facility is important during debugging. Conflict resolution is an issue only during forward inference, because backward inference does not change the memory cache.

It is important to understand how data is structured in MeriTool. MeriTool views the world as a collection of objects. There are various distinct classes of objects, each of which must be specified in an **object class definition**. During execution, there may be many individual objects of each class. For example, one may define a class of objects called ship, and then create (or otherwise consider) many different ships during execution. Each individual ship is an **instance** of the class of ships. The concept of objects is similar to that of relations (tables) in a relational data base system. A relation definition describes the structure of all tuples (rows) of that relation, and there may be any number of tuples in that relation, all of which have the same structure. For example, the ship object class is analogous to a ship relation (table) with many tuples.

Instead of columns in a relation, objects in MeriTool have **attributes**. Each attribute may be used to hold a single value, which may be of any of the valid MeriTool datatypes: **integer**, **float**, or **symbol**. An object class may have an arbitrary number of attributes, though each attribute must have a distinct name. Attribute names are "local" to each object class and, therefore, may be used within more than one object class. For example, both a ship and a plane object class could have a speed attribute.

In recognition of the internal knowledge representation approach, MeriTool is sometimes called an Object-Attribute-Value, or OAV, system.

MeriTool operates in two phases. In the first phase, a knowledge base or rule file is read and its contents are precompiled into an optimized network that facilitates efficient pattern matching and thus supports high performance execution. The network is stored in an intermediate file which is comparable to the object file created by a conventional language compiler. Because of the similarity to conventional language processing, the program which performs the first phase is sometimes called the **MeriTool rule compiler**. The rule compiler is provided as a stand-alone executable program.

The second phase loads the intermediate file into computer memory and executes the tests and actions described by the network in accordance with information

stored in the memory cache. The network-driven execution is the actual inference process. The program which performs the execution is the inference engine. The inference engine, intermediate file loader, and supporting functions make up the MeriTool application program interface (API), which is provided as an object module, to be integrated into application programs by being linked with the application.

Interprocessor Communication

All communication between the processors (PC and Silicon Graphics) is accomplished via an Ethernet hardware connection over which Transmission Control Protocol/Internet Protocol (TCP/IP) data packets are transferred. The enabling software on the Silicon Graphics is provided with the Unix(tm) operating system. On the PC, Merit used a public domain software package developed at the National Center for Supercomputing Applications and enhanced at Clarkson University. Communication between the processors is limited to the data packets defined by SeaScene (object position and orientation) and some additional initialization packets sent from the PC to the IRIS to set display configuration and the like. Communication is predominantly from the PC to the IRIS, with IRIS-to-PC communication restricted almost exclusively to acknowledgement of PC messages.

Multitasking and Interprocess Communication on the PC

The structure and complexity of the ACMEST application was best realized by providing multiple asynchronous processes on the 386PC. Unfortunately, the standard version of MS-DOS does not support multiprocessing. Rather than change operating systems (for example, to Unix) at a nontrivial monetary cost, we chose to purchase and use a multiprocessing extension to MS-DOS. The software package we selected is Desqview 386 by Quarterdeck Office Systems. Desqview supports multiple active processes under MS-DOS, timeslices the processor, manages memory to swap processes when necessary, and permits the processes to communicate with each other through a "mailbox" facility. In addition, a Desqview-aware process may spawn other processes as children. On a 386 with additional memory (such as that used for ACMEST), swapped processes can be kept in extended or expanded memory so that the swapping process is very fast -- quick enough, for example, to permit a network communication process to be managed by Desqview without loss of packets while sharing the processor with a computationally intense simulation.

In ACMEST, there are three active PC processes running under Desqview. The initialization process begins execution, spawns a simulation process and an instructional process, then becomes the network communication process which manages communication with the IRIS. Terminating ACMEST stops all of these processes.

DISCUSSION

Our success in building the ACMEST system is a demonstration of the feasibility of using rule-based inference techniques to model pilot expertise for instruction. We had identified five risk areas at the beginning of the program:

1. Knowledge acquisition,
2. Knowledge representation,
3. Validation of models (expert systems),
4. Balancing artificial intelligence (AI) and algorithmic solutions, and
5. Throughput and performance.

We were able to successfully extract knowledge of ACM tactics from existing documents (Shaw, 1985) and interviews with fighter pilots, and were also able to encode that knowledge in a form suitable for automated processing by a computer, addressing the issues in risk areas 1 and 2.

Our success in these areas was mixed, however. While we were able to develop some models of pilot expertise, we recognize that we have only begun to scratch the surface of the capabilities of victorious, surviving fighter pilots. Our models cover very few cases and the cases that are covered are very simple. We do not have an all-encompassing model of air combat decision making as was sought in the initial PRDA.

Moreover, while the models we do have are implemented and executable as inference rules on the computer, we stretched the limits of the rule-oriented representation technique. Oftentimes, we found it necessary to write rules which seem unnecessarily convoluted or strained to enable some necessary inference. In particular, many aspects of air combat seem better modeled using techniques from qualitative physics, such as confluences (de Kleer and Brown, 1984). (Qualitative physics is sometimes referred to as "naive physics" or "common sense physics".) A richer representation language than that afforded by MeriTool might have eased these problems somewhat.

We did not succeed in providing a rigorous, formal validation of either the ACM performance or the instructional appropriateness of the ACMEST system, as identified in risk area 3. In part, this failure was due to our late completion of the software, which precluded the extensive testing and evaluation on-site at Luke AFB as in the original plan. In retrospect, we should have prepared a validation plan early in the program (in the first six months), with the intent that it could change as the program evolved. If such

a plan had been available early on, we could have managed our limited resources better to balance increased functionality and validation.

The absence of formal validation does not mean that the resulting system is invalid or useless, only that we are unable to rigorously demonstrate its validity. Observers who have worked with the system have given us favorable comments and subjective evaluations have been positive. However, these noncontrolled responses provide only anecdotal evidence of acceptability and could be quite different with a slightly different audience.

We did achieve a reasonable balance between AI and conventional techniques (risk area 4). Our use of existing and commercial off-the-shelf (COTS) software provided ready-made solutions for many of the problems we might have anticipated, and left us free to focus on the unique aspects of the problem domain. The large pre-existing software base also removed the temptation to spend time creating exotic implementations of system components which were critical for operation but peripheral to the research aims of the effort.

At one point, we tipped the balance fully in favor of nonAI techniques by implementing a version of flight expertise directly in FORTRAN code. This was acceptable for a very small set of flight regimes (e.g. straight-and-level flight or sustained turns) and did provide a slight improvement in overall system responsiveness due to a lighter load on the central processing unit. However, capturing expertise directly in program source code made changes very difficult and led to problems in debugging which caused the direct programming language approach to take longer to develop than a rule-based approach. Furthermore, the direct code approach was inflexible and hard to expand or extend. While adding new rules and debugging rule sets was also more difficult than it should have been, it was definitely easier to work with rules than new programming language code.

Our concerns in risk area 5 regarding performance proved well-founded, though the system as implemented exhibits acceptable performance within the expected range. Folklore held that rule-based inference would not be able to keep up with the evolving situation, falling further and further behind as the fight progressed. We did not find that to be the case. In fact, we were pleasantly surprised that a PC-class computer running MS-DOS (admittedly a 386 with additional memory) was able to support multiple aircraft flight simulations and the rule-based models of an expert pilot, an adversary, and a coach while maintaining network interface with the Silicon Graphics computer which provided the displays.

We did do some fine tuning. We isolated the ACMEST computers from the Merit corporate Ethernet, to reduce collisions and retransmissions of the

communication packets. We converted the flight simulation from a one-process-per-aircraft version to a multiple-aircraft-in-one-process version to reduce memory use and cut down on overhead spent communicating between processes. Without these steps, there were occasional "glitches" when the simulation and display were not smooth. However, since we started the project, the speed and memory available on "desktop" PCs has grown so much that we could return to the less efficient versions on a new computer and probably still outrun the more efficient version on the existing computer.

As is often the case with research efforts, Merit believes that ACMEST has raised more questions than it has answered. In many cases, we were forced to make choices to proceed with implementation, though there was no specific reason to prefer one path over another. Our list of new and unanswered questions appears in the following paragraphs, categorized by general discipline.

Pedagogic Issues

We term issues "pedagogic" if they arise from instructional considerations.

Role in Pilot Training

As the overall force structure shrinks and training continues to increase in importance, it is necessary to address how tools like those developed under this PRDA can be integrated into existing and future training programs.

It is an open question as to where a system like ACMEST should fit in pilot training. The original announcement called for a system that would be used in transition or continuation training. In retrospect, this may not have been an appropriate location, because much of that training is concerned with switchology and cockpit procedures. Fighter tactics are certainly updated, reviewed, and practiced with new aircraft types, but the underlying issues of tactical decision making are largely established before these training programs.

ACMEST itself does not offer any guides to such decision making. Developing such guidance was not an objective of the effort, but it might be expected that some indicators of appropriate use might have emerged during development. As it happens, there are few such indicators, in part because of the broad (unfocused?) nature of the topics addressed. During development there were extended discussions about exactly what would be taught as well as how it would be taught. Occasionally we found terminology interfering with clear thought. For example, the phrase "air combat maneuvers" has a specific meaning in a progression from basic fighter maneuvers

(BFM) through dissimilar air combat tactics (DACT). The use of "ACM" in the name of the system would occasionally cause confusion through an implication that the system was intended to teach precisely that part of the progression, rather than being more concerned with tactical decision making.

We believe that the underlying approach embodied in ACMEST is applicable throughout the range of training programs, but this is only a hypothesis and would have to be established for each segment of pilot training. The success of the air intercept trainer (AIT), which shares some features with ACMEST, may suggest appropriate insertion strategies.

Once a correct training stage has been identified, the integration of an ACMEST-like system into those training processes must be addressed. The ACMEST effort produced a prototype of a tool and showed the feasibility of accomplishing certain functions through automated approaches. However, the technology development effort gives no indication of what role ACMEST or a system like it should or could play in the existing training syllabi, or what syllabus modifications would be needed to exploit the capabilities of an ACMEST-like system.

As delivered, ACMEST assumes that the student has a basic knowledge of flight procedures and can pilot an aircraft in response to direction given at the level of aircraft movement, rather than control commands (e.g. "break right" instead of "move the stick hard right"). The assumption is not enforced, i.e. -- a failure to maintain acceptable flight profiles results in a crash, but no other penalty or remedial instruction.

Structured Training vs. Free Fight

A major issue within Merit's development team was whether to make ACMEST an explicitly incremental system which progressed through exercises and skills of increasing complexity, or to provide a "free flight/fight" environment in which specific skills and techniques were developed by selecting adversary behaviors which forced the skill to be learned.

Obviously, the explicit step-by-step approach is very closely aligned with the way that training is currently performed using real aircraft and simulators. Students begin by demonstrating proficiency in simple maneuvers, proceed to more complex maneuvers, and finally complete training in an unconstrained (or minimally constrained) flight environment. Our initial work followed this approach, focusing on simple flight maneuvers and the combination of those simple maneuvers into more complex behaviors.

The other school of thought held that at least some of the reasons for the progressive approach, such as safety, did not necessarily hold in a simulated environment like ACMEST's, and that therefore we should not be wedded to existing practice. Put another way, since nothing but pride would be injured by a loss in a fight or a crash, the flight setting should always resemble full air combat as much as possible. Recognizing that learning can take place more quickly when the desired behavior is broken down into components, this approach advocated "hidden progression" by having the adversary fly in ways that made the desired behavior the appropriate choice. For example, to teach closure rate control during a tail chase, the adversary would fly in a straight line at a slower speed than the student, rather than maneuvering to gain advantage. Over time, the adversary's behavior would become more and more complex and challenging, requiring more skilled behavior on the part of the student. However, at no time would the student be explicitly told of the limitations of the adversary, and the flight environment would always appear to be unconstrained air combat, but with different levels of adversary.

The system finally delivered follows the original progressive approach. While the "unconstrained combat" approach has attractions, we chose not to follow it for three reasons:

1. It was suggested late in development when there was little time to change course. Changing courses would have put schedule and completion at risk.
2. It required a much more sophisticated adversary model, capable of determining the student's level of preparation and adjusting its flight behaviors appropriately. This level of sophistication in the adversary introduced a higher level of technical risk in an already risky program.
3. It diverged far enough from the conventional training approach to require special explanation and preparation of audiences. This increased the work required to pass on knowledge of the system and threatened acceptability.

The idea of unconstrained combat as a basic forum for fighter training remains an intriguing one and one which may be worth pursuing. In an unconstrained environment, it may be possible to place virtually all of the intelligence in the adversary, making it, in effect, an adaptive instructor pilot, and requiring the student to do all learning through guided trial and error with feedback limited to performance against the adversary. The popularity of simple flight simulation games for personal computers, with the usual "skill level" settings, argues that this approach could be quite successful.

Graphics and User Interface Issues

Issues under this heading concern the screen displays and user interaction with the ACMEST system.

Single Fixed Viewpoint/Narrow Aperture

As delivered, ACMEST routes all air combat views and student direction through the single display of a Silicon Graphics IRIS workstation. Typically, this display is a direct-view, 19-inch cathode ray tube (CRT) positioned about two to three feet from the student. This results in a relatively narrow field of view and requires explicit commands to slew the view through other angles. These display limitations of the current ACMEST system tightly restrict its ability to offer a full unconstrained air combat experience. Perhaps the most important single guideline in air combat is "keep the opponent in sight." This is nearly impossible when the student can look in only one direction and must perform nonintuitive actions (e.g. pressing keys) to change the view angle. Replacing the single CRT display with a dome, a multiCRT display, or a helmet-mounted display would mitigate these issues, but introduces other hardware requirements.

We spent some time pondering ways around the fixed/narrow viewpoint problem. While some were creative, if Rube Goldberg-like (e.g., a swivel chair with a center-mounted potentiometer to detect movement), our basic response was to note the problem and go on, leaving it for others to resolve. Based on the small amount of research we did before abandoning the topic, a low-cost helmet mounted display, like that developed at NASA Ames, seems to be the best approach to the problem at this time.

We note in passing that the viewport on the SeaScene display system can be adjusted to show as much as 120-180 degrees in a scene. Coupled with appropriate projection technology, this might offer a different approach to providing sufficient visual context to support more realistic air combat.

Interaction/Mediation Structure

As delivered, ACMEST embodies a number of design decisions regarding the user interface and the interaction between the training component of the system and the student. This particular set of decisions is not the only possible arrangement, and may not be the best possible arrangement. In particular, we identified alternatives in the following areas:

Inflight Commentary vs. Postflight Debrief. ACMEST displays text overlays on the out-the-window view whenever the student diverges from the "correct" flight path. These overlays provide instructions to recover to the desired flight path. All of the instructional interaction in ACMEST occurs during the flight, as though the instructor were looking over the student's shoulder. This running commentary or advice could be considered an interruption or distraction.

An alternative model would be a debrief, in which the student would fly an assigned mission or maneuver, and would then be debriefed after completion. We did not employ this technique since one of our goals was to provide evaluation of performance in real time during flight, but it is likely that a postflight debrief could be most beneficial, especially using the display tools, replay, and stop-action available in the ACMEST display code.

Text vs. Voice vs. Graphics. Unfortunately, text on the screen requires considerable attention for comprehension and (if not placed directly in the center of the screen) can draw the student's visual focus away from the tactical situation. We experimented briefly with synthesized speech to provide this "over the shoulder" commentary, but found the aural input to be even more distracting, though we speculate that some of the problem was due to quality issues in the voice synthesis.

Similar advisory systems which we have built in the past used large arrows to indicate the direction of desired movement rather than text. In retrospect, this probably would have been a better choice on ACMEST. Other graphical representations, such as "highway in the sky," or flight directors might offer less intrusive and more readily processed inputs to the student.

Simulation Fidelity and Display Fidelity Issues

What level of fidelity is required, in simulated behavior and displays, to achieve the desired training effect? Since the desired training effects may differ, it may be that fidelity levels will differ as well.

In our experience before this program, pilots are relatively receptive of low-to-moderate resolution visual displays (though higher quality is preferred), but are extremely critical of flight simulations which do not "feel" like the real aircraft. Nothing in our experience with ACMEST changed that assessment. Uniformly, pilots found the displays to be acceptable and the flight models to be a problem. Part of the problem with the F-16 flight model was not in the mathematics of the modeling software, but in the user interface. We offered a free-movement, two-axis joystick, where the actual aircraft has a nondeformable force stick. Since the controls were quite different (movement vs. force), the model never quite "felt right," even when the numbers matched precisely with our calibration data.

ACMEST did not address the questions of fidelity. We used existing (slightly modified) software for both simulation and display, usually accepting whatever level of fidelity that software provided. We controlled the quality of the visual representations of aircraft through our construction of visual flight models and, generally, provided what might be best characterized as "high-end, low fidelity visuals." Our simulation flight models, while reasonably fine-grained, proved difficult to tune and absorbed much of the time that we had intended to spend on issues more in the mainstream of the ACMEST effort.

Suggestions for Future Work

The following ideas are drawn from Merit's perspective. The various ideas are not exclusive -- they could be combined as desired.

1. The most obvious extension would be to increase the range of ACMEST to include a wider variety of maneuvers, counters, etc. We basically stopped when the available resources were exhausted, rather than when a completed system was achieved.
2. Validate both the expert performance and the instructional performance. Does the adversary aircraft fly competently? Does the "over-the-shoulder" expert suggest appropriate maneuvers and responses?
3. Integrate the results of other efforts into the ACMEST framework.

Specific possibilities include the results of the three other awards under this PRDA, to New Mexico State University (NMSU), Ball Systems Engineering (BSE), and Vruels Research.

The NMSU and BSE groups were developing expert pilot models using neural network approaches. There are (at least) three ways in which these models could be integrated into ACMEST.

- a. They could be used to replace the expert adversary in ACMEST.
- b. To the extent that the neural network models provide a traceable explanation of their reasoning, they might also be suitable as "glass box" experts for the instructional component.
- c. It may also be possible to use the learning components of the neural net models to derive student models, provided that there is some meaningful way to compare the resulting models with the expert model to find areas of deficiency.

Vreuls' critiquing model is ideally suited to become a core part of the instructional component. At the 1989 meeting, Vreuls indicated they expected that their component would be integrated with ours, although neither of us has pursued it.

4. Ruggedize and field the system.

At least two directions are possible here:

- a. Integrate ACMEST into the AIT hardware suite to be fielded as part of (or an enhancement to) AIT.
- b. Integrate ACMEST with a dome simulator or other means of providing a more realistic visual environment. The instructional component should stay essentially the same, but the visual would change.

In either case, it is likely that the system should be viewed as an executable requirement specification, rather than as an immediate source of program code. Taking a conservative approach would allow tighter integration with existing code and more uniform user interaction and functionality.

5. Use the system as a tool to explore training issues.

Either as a complete system or as components (simulation, visuals, instructional component), ACMEST could serve as a basis for experimentation in training. For example, an experiment could compare a staged presentation of maneuvers with the combined free-flight and remedial tutorial model.

6. Perform further research on knowledge acquisition.

NMSU and BSE are using a neural network approach to learn expert air combat behavior. There are other machine learning approaches which may also be effective as a means of acquiring expertise, such as induction and case-based learning. An effort could be undertaken to apply different learning techniques to

the same training data, and then compare the performance of the learned systems against both human experts and each other. Other evaluation criteria might include the ability to use the resulting expertise models in an instructional setting.

REFERENCES

- de Kleer, J., & Brown, J.S. (1984). A qualitative physics based on confluences. Artificial Intelligence, 24. Republished in D.G. Bobrow (Ed.) (1985), Qualitative Reasoning about Physical Systems. Cambridge: MIT Press.
- Forgy, C.L. (1979). On the Efficient Implementation of Production Systems. Unpublished doctoral dissertation, Carnegie-Mellon University, Pittsburgh, PA.
- Frost, R. (1986). Introduction to Knowledge Base Systems. New York: Macmillan.
- Shaw, R.L. (1985). Fighter Combat: Tactics and Maneuvering. Annapolis: Naval Institute Press.

APPENDIX: CHRONOLOGY

July 1988

Contract awarded (29 July).

August 1988

Kickoff meeting at Williams AFB, AZ.

Established consulting agreement with Fighter Command International.

Created ACMEST team at Merit, assembled computing resources.

Started knowledge acquisition.

September 1988

Began assembling existing code components.

Continued knowledge engineering.

October 1988

Meeting with Fighter Command International (FCI) to discuss training procedures, sequences, and problems. Focus on basic fighter maneuvers (BFM) and collected "talk-throughs" of standard maneuvers. Also collected several pilot training and evaluation documents.

Identified single fixed display as a potential problem area.

Prepared for principal investigator's meeting at Williams AFB in November.

November 1988

Participated in principal investigator's meeting at Williams AFB.

Flight simulation software successfully ran one aircraft on a PC-class computer.

Decided to focus first on 1v0 maneuvers, since their correctness can be judged using laws of physics rather than tactical evaluation.

December 1988

Reviewed documentation on Advanced Maneuvering Logic (AML) and TAC Brawler.

Continued to work on PC-based flight simulation (multiple aircraft).

Requested System Specification for the Performance Measurement System.

January 1989

Prepared a Knowledge Engineering Plan to structure the acquisition of pilot expertise.

Fixed some problems with the PC-based multiple aircraft flight simulation, and started designing the interface between the PC and a Silicon Graphics-based display program.

February 1989

Devised a multi-level option tree representation for pilot expertise, using a simple situation as a test case.

Created rule sets from AML and TAC Brawler literature for evaluation.

March 1989

Filled in the option tree representation of pilot expertise, and started to implement it as a collection of rules.

Tested AML and TAC Brawler-derived rule sets, with little satisfaction. Neither appears suitable as either a model of expertise or adversary control.

Received a copy of the System Specification for the Performance Measurement System (PMS) at Luke AFB.

April 1989

Moved a flat-earth, out-the-window view graphics package onto the Silicon Graphics workstation being used for development, and started the process of interfacing the graphics package with the simulation running on the 80386 PC.

Received some PMS data which we intend to use as a "sanity check" during instructional system development.

Reviewing the PMS System Specification in preparation for interface design.

May 1989

Adapting some public domain Ethernet software for inter-computer communication between the 80286 PC running the simulation software and the Silicon Graphics workstation running the display software. The display and simulation software components are each operating independently and the network software was supporting communication between computers, but the communication had not yet been integrated into the support software packages.

June 1989

Completed the adaptation of public domain Ethernet software. At the end of June, the network software was supporting communication between computers, allowing the simulation to drive the display directly.

Began work on three-dimensional display models of the various aircraft types needed to support ACMEST. This work is being performed by a summer hire college student.

Developing accurate performance models of the various aircraft, working from available nonproprietary and unclassified sources. These performance models are used by the simulation to drive the flight models.

Preparing a letter which will seek clarification on some points raised by our study of the PMS system specification document.

July 1989

Successfully flying on our 386 PC and Personal Iris computer configuration, with 3D F-15 and F-16 icons, an F-16 HUD, and an F-16 dynamics model.

Prepared for and attended the annual project review at Williams AFB on 27 July, along with the other contractors working under this PRDA.

Asked our consulting pilot, Mr. Robert Shaw, to keep a journal and collect other information while undergoing F-16 transition training in Wichita for two months beginning in mid-August.

Anticipating a visit by AL/HRAU in September or October to review the implementation.

August 1989

The principal implementor attended a Desqview developer's workshop for three days in August, to better use the capabilities of the multitasking system to support ACMEST.

September 1989

Added an expert system controller to the program configuration on the 386 host computer. The base simulation software also runs on this computer, and multitasking software is used to share the computing resource among tasks.

Scheduled a visit by Dr. Richard Thurman of AL/HRAU on October 26. We will be discussing our progress to date, our plans, and providing demonstrations of the existing system capabilities.

October 1989

Enhanced the expert system controller to control and coach a level sustained turn in addition to straight-and-level flight.

Corrected a number of minor weaknesses in the underlying simulation, display, and communications software made apparent by the expert system controller.

Enjoyed a brief visit from Dr. Richard Thurman of AL/HRAU, and appreciated the opportunity to give him a better understanding of our plans, goals, and current progress.

November 1989

Principal ACMEST software developer resigned from Merit.

Our consulting pilot, Robert Shaw, president of Fighter Command International and author of Fighter Combat: Tactics and Maneuvering, visited to critique our flight dynamics model, to brief us on F-16 transition training (which he had just completed), and to advise us on other aspects of the ACMEST system. He identified significant problems with the F-16 flight model which will require repair.

December 1989

Identified the sources of instability in the flight model. Corrections have been made and are being tested before insertion into the system.

Our college student returned over his winter break and has added elevation view and screen configuration capabilities to the display functions.

Hired a replacement for the departed principal developer.

January 1990

Ascertained that a significant reason for lack of responsiveness in the flight model is heavy network traffic on the link between the simulation computer and the display computer.

Modifying our planned user interaction based on recommendations from our pilot/instructor consultant.

Had some success in difference detection based on simple comparison of control inputs. Investigating the use of an intent hierarchy to overcome limitations on control input difference detection.

Increasingly finding it necessary to introduce convoluted and opaque constructions in the rule language to produce desired behaviors.

Two new personnel joined the program.

February 1990

Isolated the ACMEST computers on an independent Ethernet loop and saw the expected improvement in responsiveness of the flight model.

Our fighter pilot consultant visited and commented that the improved flight model was much better.

Working on the student modelling and difference detection problems.

March 1990

Replaced the Personal Iris (4D/25) with an Iris model 3030 in our hardware configuration.

Spent some time resolving problems with the flight model.

Working on the adversary and student models in parallel. Implementing the adversary model as rule bases. Because of the requirement for change and update, we have not yet committed to implementing the student model as rules.

Added Dr. Marshall Holman to the team on a part-time basis for guidance in training issues and help in the preparation of the demonstration plan.

April 1990

Resolved most of the problems with the flight model and have multiple aircraft flying under manual and/or automatic control.

Completed a plan detailing the performance of the ACMEST system to be demonstrated at the Second Annual Review.

Spent some time refining our instructional components of the system.

Designed the system/user interface.

May 1990

Revised the preliminary demonstration plan.

Improved the overall system throughput by converting from separate processes for each aircraft to a single process simulating all aircraft.

Implemented the primary flight procedure components that are necessary for the August demonstration, including pure pursuit, sustained turn, straight-and-level flight, and lead pursuit.

Due to a reorganization at Merit, Bob Bechtel was replaced as program manager by Sid Sipes.

June 1990

Working on detecting student errors. We are focusing on errors in speed management for the August demonstration.

Correcting minor problems to support August demonstration.

July 1990

Developed and integrated a view direction indicator. Provided both a manual view change function, where the pilot can select a view using a single keypress, and an automated "tracking" view in which the view direction follows the adversary aircraft. Made some additions to the three-dimensional aircraft visual models so that viewing is realistically obstructed by the aircraft surfaces.

Implemented an instructional management component and user interface for tracking students and their progress.

Revised and improved the message set for intermachine communication.

August 1990

Demonstrated (20 August) on-screen corrective advice to a student flying a lead pursuit intercept on an adversary flying a sustained turn. Also shown was the skeleton of an instructor record-keeping system for tracking students, classes, and lessons.

Discussed ACMEST to date and future plans with AL/HRAU following the demonstration.

September 1990

Followed up on three issues raised by the August status review and status report - focus of remaining efforts, hardware issues, and assessment procedures.

Efforts will be minimized until FY91 funding is received.

October 1990

Currently authorized funds had been expended so efforts were essentially shut down.

November 1990

Received authorization to expend the remaining funds to total contract value.

December 1990

Worked with Robert Shaw to identify instructor-level concerns which should be reflected in an evaluation of ACMEST.

Examined the existing demonstration system to plan the expenditure of the remaining resources to reach the planned deliveries.

January 1991

Little effort (2 hours) was expended in January due to a lack of personnel availability.

February 1991

Little effort (2 hours) was expended due to a lack of personnel availability.

March 1991

Little effort (1 hour) was expended due to a lack of personnel availability. The requirement for personnel to complete the program (software delivery and final report) was raised to both the Vice President and President of Merit.

April 1991

Prepared both a software version description document and a hardware requirements list.

May 1991

Submitted hardware requirements list and planned software/GFE delivery list.

June 1991

Established a delivery and installation schedule.

July 1991

The final report was delayed by a lack of personnel resource at Merit. Working to resolve this situation.

August 1991

Delivered first submission of final report.

Shipped software.